ELSEVIER

Information Sciences 139 (2001) 59–77

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A new algorithm for computing similarity between RNA structures ☆

Gregory D. Collins [a,1], Shuyun Le [b], Kaizhong Zhang [a,*]

[a] *Department of Computer Science, University of Western Ontario, London, Ont., Canada, N6A 5B7*

[b] *Laboratory of Experiments and Computer Biology, National Cancer Institute, NIH, Frederick, MD 21702, USA*

Received 10 September 1999; received in revised form 24 February 2001; accepted 1 April 2001

Abstract

The primary structure of a ribonucleic acid (RNA) molecule is a sequence of nucleotides (bases) over the four-letter alphabet $\{A, C, G, U\}$. The secondary or tertiary structure of an RNA is a set of base-pairs (nucleotide pairs) which form bonds between A–U and C–G. For secondary structures, these bonds have been traditionally assumed to be one-to-one and non-crossing. We consider the edit distance between two RNA structures. This is a notion of similarity, introduced in [Proceedings of the Tenth Symposium on Combinatorial Pattern Matching, Lecture Notes in Computer Science, vol. 1645, Springer, Berlin, 1999, p. 281], between two RNA molecule structures taking into account the primary, the secondary and the tertiary structures. In general this problem is NP-hard for tertiary structures. In this paper, we consider this notion under some constraints. We present an algorithm and then show how to use this algorithm for practical applications. © 2001 Elsevier Science Inc. All rights reserved.

Keywords: Molecular biology; RNA structures; Similarity

☆ Research supported partially by the Natural Sciences and Engineering Research Council of Canada under Grant No. OGP0046373.

* Corresponding author.

E-mail addresses: gcollins@cs.yale.edu (G.D. Collins), shuyun@ncifcrf.gov (S. Le), kzhang@csd.uwo.ca (K. Zhang).

[1] Present address: Department of Computer Science, Yale University, P.O. Box 208285, New Haven, CT 06520-8285, USA.

0020-0255/01/$ - see front matter © 2001 Elsevier Science Inc. All rights reserved.
PII: S0020-0255(01)00157-8

1. Introduction

Ribonucleic acid (RNA) is an important molecule which performs a wide range of functions in biological systems. In particular, it is RNA (not DNA) that contains the genetic information of viruses such as HIV, and therefore describes the effect of the functions of such viruses. RNA has recently become the center of much attention because of its catalytic properties, leading to an increased interest in obtaining structural information.

It is well known that secondary and tertiary structural features of RNAs are important in the molecular mechanism involving their functions. The presumption, of course, is that to a preserved function there corresponds a preserved molecular confirmation and, therefore, a preserved secondary and tertiary structure. Therefore the ability to compare RNA structures is useful [1,3,4,6,7,9,10].

In an RNA secondary or tertiary structure, a bonded pair of bases (base-pair) is usually represented as an edge between the two complementary bases involved in the bond. It is assumed that any base participates in at most one such pair. The RNA secondary structure is defined such that the edges of the bonded pairs are non-crossing.

In [14,16], a similarity measure between RNA structures was introduced. First three basic operations, insertion, deletion, and relabel, were defined on base pairs and single bases. The measure between two RNAs is then defined to be the minimum number of (weighted) operations that can transform one RNA into the other. If both structures are secondary, then using the solution in [14], we can compute the optimal solution using ordered tree edit distance algorithms [5,15]. If at least one of the structures is secondary, then using the solution in [16], one can find the optimal solution. When both structures are tertiary, then the problem is NP-hard [16].

In this paper, we consider this problem under some constraints. Given two RNAs R_1 and R_2, a set of base pairs $\{p_{1_1}, \ldots, p_{1_K}\}$ from R_1, and a set of base pairs $\{p_{2_1}, \ldots, p_{2_K}\}$ from R_2, we would like to find the minimum number of (weighted) operations that can transform R_1 into R_2 under the condition that p_{1_i} has be matched to p_{2_i} for some $1 \leqslant i \leqslant K$.

We will present an algorithm for this problem. We will show that this algorithm can be used to accommodate the situation where both structures are tertiary structures.

2. Comparing two RNA structures

2.1. RNA structures and edit distance

The primary structure of an RNA molecule is a sequence of nucleotides (bases) over the four-letter alphabet $\Sigma = \{A, C, G, U\}$. The secondary or ter-

tiary structure of an RNA is a set of base-pairs (nucleotide pairs) which form bonds between A–U and C–G. Following Zuker [17–19], we assume a model where there are no "knots" in the secondary structure. This means that for the secondary structure, the bonds are non-crossing. For tertiary structures, there is no non-crossing restriction.

An RNA structure is represented by $R(P)$, where R is a sequence of nucleotides and $P \subseteq \{1,2,\ldots,|R|\}^2$ is a set of pairs of which each element (i,j) represents a base pair $(R[i],R[j])$. We use $R[i]$ to represent the ith nucleotide of R. We use $R[i..j]$ to represent the sequence of nucleotides from $R[i]$ to $R[j]$. We assume that base pairs in R do not share participating bases. Formally for any (i_1,j_1) and (i_2,j_2) in P:

$$j_1 \neq i_2, \quad i_1 \neq j_2, \quad \text{and} \quad i_1 = i_2 \text{ if and only if } j_1 = j_2.$$

Given an RNA structure $R(P)$, we use $S(R,P)$ to represent the set of structural elements consisting of both its set of base-pairs and the remaining unpaired nucleotides.

$$S(R,P) = P \cup \{(i,i) \,|\, R[i] \text{ is not involved in any base pair in } R\}.$$

We use $S(R,P)[i..j]$ to represent the set of structural elements in sequence $R[i..j]$.

$$S(R,P)[i..j] = \{r \,|\, r = (k,l) \in S(R,P),\ i \leqslant k,\ l \leqslant j\}.$$

For $r = (i,j) \in S(R,P)$, we use $\text{label}_R(r)$ to represent the label of r in R. If $i = j$, then $\text{label}_R(r) = R[i] = R[j]$, otherwise $\text{label}_R(r) = R[i]R[j]$. For $r = (i,j) \in S(R,P)$, i and j are often called the $5'$ end and $3'$ end of r, respectively. We define $\text{left}(r) = i$ and $\text{right}(r) = j$.

Following the tradition in sequence comparison [8,11,12], we define three operations, relabel, delete, and insert, on RNA structures. For a given RNA structure R, each operation can be applied to either a base-pair in $S(R,P)$ or an unpaired base. Relabelling a base-pair is to replace one base-pair in $S(R,P)$ with another. This means that at the sequence level, two bases may be changed at the same time. Deleting a base-pair is to delete the pair from $S(R,P)$, removing two bases from the sequence at the same time. Inserting a base-pair is to insert a new base-pair into $S(R,P)$, introducing two bases into the sequence at the same time. Relabelling an unpaired base is to replace it with another base. Deleting an unpaired base is to delete the base from the sequence. Inserting a base is to insert a new base into the sequence as an unpaired base. Note that there is no relabel operation that can change a base-pair to an unpaired base or vice versa.

The assumption here is that a base-pair is a whole entity. One cannot delete one base of a base-pair and change the other base of the same base-pair. This assumption seems fit better with the comparative analysis method used either manually or automatically by biologists.

Following [13,15,16], we represent an edit operation as $a \to b$, where a and b are either λ (the empty structural element) or they are both labels of base-pairs from $\{A, C, G, U\} \times \{A, C, G, U\}$ or unpaired bases from $\{A, C, G, U\}$.

We call $a \to b$ a change operation if $a \neq \lambda$ and $b \neq \lambda$; a delete operation if $b = \lambda$; and an insert operation if $a = \lambda$.

Let S be a sequence $s_1, \ldots, s_k$ of edit operations. An S-derivation from RNA structure A to RNA structure B is a sequence of RNA structures $A_0, \ldots, A_k$ such that $A = A_0$, $B = A_k$, and $A_{i-1} \to A_i$ via s_i for $1 \leqslant i \leqslant k$.

Let γ be a cost function which assigns to each edit operation $a \to b$ a nonnegative real number $\gamma(a \to b)$. We constrain γ to be a distance metric. That is:

- $\gamma(a \to b) \geqslant 0, \quad \gamma(a \to a) = 0$;
- $\gamma(a \to b) = \gamma(b \to a)$; and
- $\gamma(a \to c) \leqslant \gamma(a \to b) + \gamma(b \to c)$.

We extend γ to a sequence of edit operations $S = s_1, \ldots, s_k$ by letting $\gamma(S) = \sum_{i=1}^{|S|} \gamma(s_i)$.

The *edit distance* between two RNA structures is defined by considering the minimum cost edit operation sequence that transforms one structure to the other. Formally the edit distance between $R_1(P_1)$ and $R_2(P_2)$ is defined as:

$$D(R_1(P_1), R_2(P_2)) = \min_T \{\gamma(T) \mid T \text{ is an edit operation sequence taking } S(R_1, P_1) \text{ to } S(R_2, P_2)\}.$$

2.2. Mapping between RNA structures

Let $r = (r_l, r_r)$ and $s = (s_l, s_r)$ be two elements in $S(R, P)$ of an RNA $R(P)$. We define the relation between r and s as follows. We say r is *before* s if $r_r < s_l$. We say r is *cross-before* s if $r_l < s_l < r_r < s_r$. We say r is *inside* s if $s_l < r_l$ and $r_r < s_r$. Fig. 1 gives an illustration of these concepts.

Let $R_1(P_1)$ and $R_2(P_2)$ be two RNA structures. Formally we define a triple (M, R_1, R_2) to be a mapping from $R_1(P_1)$ to $R_2(P_2)$, where M is a binary relation on $S(R_1, P_1) \times S(R_2, P_2)$ such that

1. For any (r, s) in M,
   r is a base-pair in P_1 if and only if s is a base-pair in P_2.

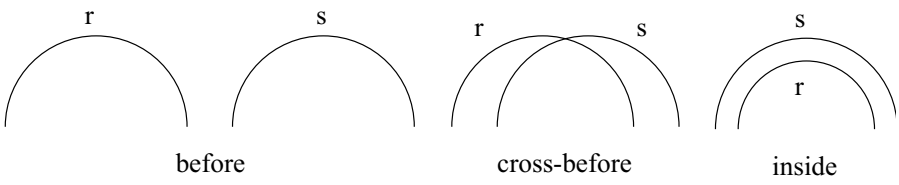

Fig. 1. Relationship between r and s.

2. For any pair of (r_1, s_1) and (r_2, s_2) in M,
   (a) $r_1 = r_2$ if and only if $s_1 = s_2$ (M is one-to-one).
   (b) r_1 is *before* r_2 if and only if s_1 is *before* s_2.
   (c) r_1 is *inside* r_2 if and only if s_1 is *inside* s_2.
   (d) r_1 is *cross_before* r_2 if and only if s_1 is *cross_before* s_2.

We will use M instead of (M, R_1, R_2) if there is no confusion. Let M be a mapping from $R_1(P_1)$ to $R_2(P_2)$. Then we can similarly define the cost of M:

$$\gamma(M) = \sum_{(r,s)\in M} \gamma(\text{label}_{R_1}(r) \to \text{label}_{R_2}(s)) + \sum_{r\notin M} \gamma(\text{label}_{R_1}(r) \to \lambda) + \sum_{s\notin M} \gamma(\lambda \to \text{label}_{R_2}(s)).$$

Mappings can be composed. Let M_1 be a mapping from $R_1(P_1)$ to $R_2(P_2)$ and M_2 be a mapping from $R_2(P_2)$ to $R_3(P_3)$. Define

$$M_1 \circ M_2 = \{(r,t) \mid \exists s \quad \text{s.t. } (r,s) \in M_1 \text{ and } (s,t) \in M_2\}.$$

Lemma 1.
1. *$M_1 \circ M_2$ is a mapping between $R_1(P_1)$ and $R_3(P_3)$.*
2. *$\gamma(M_1 \circ M_2) \leqslant \gamma(M_1) + \gamma(M_2)$.*

Proof. From [16]. □

The relation between a mapping and a sequence of edit operations is as follows.

Lemma 2. *Given S, a sequence $s_1, \ldots, s_k$ of edit operations from $R_1(P_1)$ to $R_2(P_2)$, there exists a mapping M from $R_1(P_1)$ to $R_2(P_2)$ such that $\gamma(M) \leqslant \gamma(S)$. Conversely, for any mapping M, there exists a sequence of edit operations such that $\gamma(S) = \gamma(M)$.*

Proof. From [16]. □

Based on the lemma, the following theorem states the relation between the distance and the mappings.

Theorem 1.
$D(R_1(P_1), R_2(P_2)) = \min_M\{\gamma(M) \mid M \text{ is a mapping from } R_1(P_1) \text{ to } R_2(P_2)\}$.

Proof. From [16]. □

2.3. Constrained edit distance

We now consider the edit distance between two RNAs with constraints. Given $R_1(P_1)$ and $R_2(P_2)$ and a mapping M between them, we would like to find

the edit distance between them with the condition that for any $(r,s) \in M$, r is constrained to match to s.

Formally given $R_1(P_1)$ and $R_2(P_2)$ and a mapping M between them, we define the constrained edit distance between $R_1(P_1)$ and $R_2(P_2)$ with constraint M as follows:

$$D(R_1(P_1), R_2(P_2), M) = \min_{M'}\{\gamma(M') \mid M' \text{ is a mapping from } R_1(P_1) \text{ to } R_2(P_2) \text{ and } M \subseteq M'\}.$$

The following lemma is immediate from this definition and Theorem 1.

Lemma 3.

$$D(R_1(P_1), R_2(P_2)) = D(R_1(P_1), R_2(P_2), \emptyset),$$
$$D(R_1(P_1), R_2(P_2)) \leqslant D(R_1(P_1), R_2(P_2), M).$$

Proof. Trivial. □

We consider the properties of mappings. Suppose that M is a mapping between $R_1(P_2)$ and $R_2(P_2)$. We define a binary relation S_M on $\{0,1,\ldots,|R_1|+1\} \times \{0,1,\ldots,|R_2|+1\}$ as follows:

$$S_M = \{(\text{left}(r), \text{left}(s)), (\text{right}(r), \text{right}(s)) \mid (r,s) \in M\} \cup \{(0,0), (|R_1|+1, |R_2|+1)\}.$$

Lemma 4. *Given a mapping M between $R_1(P_2)$ and $R_2(P_2)$, then S_M satisfies the following condition. For any (i_1, j_1) and (i_2, j_2) in S_M,*
- *$i_1 = i_2$ if and only if $j_1 = j_2$,*
- *$i_1 < i_2$ if and only if $j_1 < j_2$,*

Proof. Immediate from the definition of mappings. □

From this lemma, S_M can be represented as a sequence of ordered pairs as follows, where $i_k < i_{k+1}$, $j_k < j_{k+1}$ for $0 \leqslant k < K$:

$$S_M = ((i_0 = 0, j_0 = 0), (i_1, j_1), (i_2, j_2), \ldots, (i_K = |R_1| + 1, j_K = |R_2| + 1)).$$

Let (r,s) be a pair in a mapping M' from $R_1(P_1)$ to $R_2(P_2)$, We say (r,s) is *compatible* with M or S_M if there exist k and l such that
- $i_k < \text{left}(r) < i_{k+1}$,
- $j_k < \text{left}(s) < j_{k+1}$,
- $i_l < \text{right}(r) < i_{l+1}$,
- $j_l < \text{right}(s) < j_{l+1}$.

Lemma 5. *Suppose that M is a mapping between $R_1(P_1)$ and $R_2(P_2)$, and $S_M = ((i_0, j_0), (i_1, j_1), \ldots, (i_K, j_K))$. Let M' be another mapping between $R_1(P_1)$*

and $R_2(P_2)$ such that $M \cap M' = \emptyset$, then $M \cup M'$ is a mapping if and only if for any $(r,s) \in M'$, (r,s) is compatible with S_M.

Proof. Suppose that for any $(r,s) \in M'$, (r,s) is compatible with S_M. Let (r_1,s_1) and (r_2,s_2) be in $M \cup M'$, we want to show that they satisfy the mapping condition. We will only consider the case where (r_1,s_1) is in M and (r_2,s_2) is in M' since M and M' are both mappings. We now assume that r_1 and r_2 are both base pairs, and therefore s_1 and s_2 are also base pairs. Other cases where either r_1 or r_2 is a single base are easier to analyse and are omitted for brevity.

Since (r_1,s_1) is in M, there exist k and l such that $i_k = \text{left}(r_1)$, $i_l = \text{right}(r_1)$, $j_k = \text{left}(s_1)$, and $j_l = \text{right}(s_1)$. By the definition of compatible, we have $\text{left}(r_2) < i_t$ if and only if $\text{left}(s_2) < j_t$, and $\text{right}(r_2) < j_t$ if and only if $\text{right}(s_2) < j_t$, where $t = k, l$. This means that the relation between r_1 and r_2 and the relation between s_1 and s_2 are the same. Therefore $M \cup M'$ is a mapping.

On the other hand, suppose that $M \cup M'$ is a mapping, but there is a $(r,s) \in M'$ such that (r,s) is not compatible with S_M. Then there is a k such that either $\text{left}(r) < i_k$ and $\text{left}(s) > j_k$ or $\text{right}(r) < i_k$ and $\text{right}(s) > j_k$. Note that we do not consider the cases where, for some k, $\text{left}(r) = i_k$, $\text{left}(s) = j_k$, $\text{right}(r) = i_k$ or $\text{right}(s) = j_k$. The reason for this is that base pairs do not share end bases and $M \cap M' = \emptyset$. This means that there is some $(r_1,s_1) \in M$ such that (r,s) and (r_1,s_1) do not satisfy mapping conditions. Therefore $M \cup M'$ is not a mapping which is a contradiction. □

3. Algorithms

In this section, we assume that $R_1(P_1)$, $R_2(P_2)$, and M are given and $S_M = ((i_0,j_0),(i_1,j_1),\ldots,(i_K,j_K))$.

Let $M_1 = \{s = (l,r) \mid l \neq r \text{ and } \exists t \text{ such that } (s,t) \in M\}$, and $M_2 = \{t = (l,r) \mid l \neq r \text{ and } \exists s \text{ such that } (s,t) \in M\}$. We now consider the case where at most one of $P_1 - M_1$ and $P_2 - M_2$ is a tertiary structure. We present an algorithm which solves this problem. Our algorithm can be used for comparing tertiary structures in practical application.

3.1. Properties

We use a bottom–up approach. We consider smaller substructures first and eventually consider the whole structure.

We first consider how the given mapping M will limit the substructures we have to consider. We say i and j are *consistent* with respect to S_M if there is a k such that either $i = i_k$ and $j = j_k$ or $i_k < i < i_{k+1}$ and $j_k < j < j_{k+1}$.

For any $D(R_1[l_1..r_1], R_2[l_2..r_2])$ such that either l_1 and l_2 are not consistent or r_1 and r_2 are not consistent, there exists a k such that either $l_1 \leqslant i_k \leqslant r_1$ but

$j_k < l_2$ or $j_k > r_2$, or $l_2 \leqslant j_k \leqslant r_2$ but $i_k < l_1$ or $i_k > r_1$. In both cases, $D(R_1[l_1..r_1], R_2[l_2..r_2])$ is not useful since one of the matching elements in M has to be deleted.

We can now consider how to compute $D(R_1[l_1..r_1], R_2[l_2..r_2])$, where (l_1, l_2) and (r_1, r_2) are consistent with respect to S_M.

Let $S_1[1..m]$ and $S_2[1..n]$ be arrays containing pairs in $S(R_1, P_1 - M_1)[l_1..r_1]$ and $S(R_2, P_2 - M_2)[l_2..r_2]$, sorted by $3'$ end.

Let $S_1[i] = (s_1, t_1)$ and $S_2[j] = (s_2, t_2)$, we define $\text{left}_1[i]$, $\text{cross\_left}_1[i]$ and $\text{cross\_weight}_1[i]$ as follows; $\text{left}_2[j]$, $\text{cross\_left}_2[j]$ and $\text{cross\_weight}_2[j]$ are defined similarly.

$$\text{left}_1[i] = \begin{cases} \max\{k\} & S_1[k]'s\ 3' \text{ end is less than } s_1, \\ 0 & \text{if no such } k \text{ exists}, \end{cases}$$

$$\text{cross\_left}_1[i] = \begin{cases} 1 & \text{if there exists a } k < i \text{ such that} \\ & \quad S_1[k] \text{ is cross-before } S_1[i], \\ 0 & \text{if no such } k \text{ exists}, \end{cases}$$

$$\text{cross\_weight}_1[i] = \sum_{1 \leqslant k < i, S_1[k] \text{ is cross-before } S_1[i]} \gamma(\text{label}_{R_1}(S_1[k]) \to \lambda).$$

Again, let $S_1[i] = (s_1, t_1)$ and $S_2[j] = (s_2, t_2)$. We now define $D_1(i, j)$ and $D_2(i, j)$ as follows.

$$D_1(i, j) = D(R_1[l_1..t_1], R_2[l_2..t_2], M),$$
$$D_2(i, j) = D(R_1[s_1..t_1], R_2[s_2..t_2], M).$$

Lemma 6. *Suppose that $i = i_k$ and $j = j_k$, then*

$$D_1(i, j) = D_1(i - 1, j - 1).$$

Proof. As we require that M be a part of the mapping between $R_1(P_1)$ and $R_2(P_2)$, the actual costs of elements in M do not affect the final solution. Therefore we can pre-calculate the cost of the constraints and assume that the cost is zero in our algorithm. □

Lemma 7. *Suppose that $i = i_k$ and $j_k < j < j_{k+1}$, then*

$$D_1(i, j) = D_1(i, j - 1) + \gamma(\lambda \to \text{label}_{R_2}(S_2[j])).$$

Proof. In this case, since i_k must match to j_k, $S_2[j]$ has to be inserted. □

Lemma 8. *Suppose that $i_k < i < i_{k+1}$ and $j = j_k$, then*

$$D_1(i, j) = D_1(i - 1, j) + \gamma(\text{label}_{R_1}(S_1[i]) \to \lambda).$$

Proof. In this case, since i_k must match to j_k, $S_1[i]$ has to be deleted. □

In the rest of this section, we assume that there is no k such that $i = i_k$ or $j = j_k$.

Lemma 9. *Suppose that $S_1[i]$ is a single base and $S_2[j]$ is a base pair or vice versa, then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(\text{label}_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to \text{label}_{R_2}(S_2[j])). \end{cases}$$

Proof. Since a single base cannot be matched to a base pair, we must insert or delete either the single base or the base pair. □

Lemma 10. *Suppose that $S_1[i]$ and $S_2[j]$ are both single bases, then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(\text{label}_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to \text{label}_{R_2}(S_2[j])), \\ D_1(i-1,j-1) + \gamma(\text{label}_{R_1}(S_1[i]) \to \text{label}_{R_2}(S_2[j])). \end{cases}$$

Proof. In this case, one can either delete one of the single bases or match them together. □

Lemma 11. *Suppose that $S_1[i]$ and $S_2[j]$ are both base pairs and $S_1[i]$ and $S_2[j]$ are not compatible with S_M, then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(\text{label}_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to \text{label}_{R_2}(S_2[j])). \end{cases}$$

Proof. In this case, one can delete either $S_1[i]$ or $S_2[j]$ because matching $S_1[i]$ with $S_2[j]$ will violate mapping conditions. □

Lemma 12. *Suppose that $S_1[i]$ and $S_2[j]$ are both base pairs and $S_1[i]$ and $S_2[j]$ are compatible with S_M. If* $\text{left}_1[i] \neq 0$, $\text{left}_2[j] \neq 0$, $\text{cross\_left}[i] \neq 0$, *or* $\text{cross\_left}[j] \neq 0$, *then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(\text{label}_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to \text{label}_{R_2}(S_2[j])), \\ D_1(\text{left}_1[i], \text{left}_2[j]) + D_2(i,j) + \text{cross\_weight}_1[i] \\ \quad + \text{cross\_weight}_2[j]. \end{cases}$$

Proof. Let $S_1[i] = (s_1, t_1)$ and $S_2[j] = (s_2, t_2)$. Consider the best mapping between $R_1[l_1..t_1]$ and $R_2[l_2..t_2]$. If $S_1[i] = (s_1, t_1)$ is not in the mapping, then

$$D_1(i,j) = D_1(i-1,j) + \gamma(\text{label}_{R_1}(S_1[i]) \to \lambda).$$

If $S_2[j] = (s_2, t_2)$ is not in the mapping, then

$$D(i,j) = D_1(i,j-1) + \gamma(\lambda \to \text{label}_{R_2}(S_2[j])).$$

If both $S_1[i] = (s_1, t_1)$ and $S_2[j] = (s_2, t_2)$ are in the mapping, then they should map to each other by the definition of mapping. In this case, since one of the structures is a secondary structure, any base pair cross-before $S_1[i]$ or $S_2[j]$ will not be in the mapping and should be deleted. Therefore, if $\text{left}_1[i] \neq 0$ or $\text{left}_2[j] \neq 0$,

$$D(i,j) = D_1(\text{left}_1[i], \text{left}_2[j]) + D_2(i,j) + \text{cross\_weight}_1[i] + \text{cross\_weight}_2[j].$$

If $\text{left}_1[i] = 0$ and $\text{left}_2[j] = 0$, and $\text{cross\_left}[i] \neq 0$, or $\text{cross\_left}[j] \neq 0$, then

$$D(i,j) = D_2(i,j) + \text{cross\_weight}_1[i] + \text{cross\_weight}_2[j].$$

If we define $D(0,0) = 0$, then we can combine the above two cases. Note that one of the cross-weights is zero since in secondary structures there is no crossing. Also if $S_1[i]$ and $S_2[j]$ are both single bases, both cross-weights are zero. □

Lemma 13. *Suppose that $S_1[i]$ and $S_2[j]$ are both base pairs and $S_1[i]$ and $S_2[j]$ are compatible with S_M. If* $\text{left}_1[i] = 0$, $\text{left}_2[j] = 0$, $\text{cross\_left}[i] = 0$, *and* $\text{cross\_left}[j] = 0$, *then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(\text{label}_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to \text{label}_{R_2}(S_2[j])), \\ D_1(i-1,j-1) + \gamma(\text{label}_{R_1}(S_1[i]) \to \text{label}_{R_2}(S_2[j])). \end{cases}$$

Proof. Let $S_1[i] = (s_1, t_1)$ and $S_2[j] = (s_2, t_2)$. Consider the best mapping between $R_1[l_1..t_1]$ and $R_2[l_2..t_2]$. The first two cases are similar to Lemma 12. For the last case, since there is no pair before or cross-before $S_1[i]$ or $S_2[j]$, $S_1[k]$, $1 \leqslant k < i$, is inside $S_1[i]$ and $S_2[k]$, $1 \leqslant k < j$, is inside $S_2[j]$. Therefore $D_1(i,j) = D_1(i-1,j-1) + \gamma(\text{label}_{R_1}(S_1[i]) \to \text{label}_{R_2}(S_2[j]))$. □

3.2. Algorithm

From the above lemmas, we can compute $D(R_1(P_1), R_2(P_2), M)$ using a bottom-up approach. Since we only use $D_2(i,j)$ in Lemma 12, we only need to compute these $D(R_1[l_1..r_1], R_2[l_2..r_2], M)$ such that $(l_1, r_1) \in P_1$ and $(l_2, r_2) \in P_2$. Furthermore, by Lemma 13, we can reduce the computation from each pair of base pairs to each pair of stems, where a stem in an RNA is a set of stacked pairs of maximum size.

Given $R_1(P_1)$ and $R_2(P_2)$, we can first compute sorted stem lists L_1 for R_1 and L_2 for R_2. It follows from the above discussion that, for each pair of stems $L_1[i] = (l_1, r_1, k_1)$ and $L_2[j] = (l_2, r_2, k_2)$ such that (l_1, l_2) and (r_1, r_2) are consistent with S_M, we have to compute $D(R_1[l_1, r_1], R_2[l_2, r_2], M)$. Fig. 3 shows the algorithm. We use Lemmas 9–13 to compute $D(R_1[l_1, r_1], R_2[l_2, r_2], M)$. Fig. 2 shows this computation.

Let $R_1[1..m](P_1)$ and $R_2[1..n](P_2)$ be the two given RNA structures. Let $\text{stem}(R_1)$ and $\text{stem}(R_2)$ be the number of stems in R_1 and R_2, respectively.

Input: $R_1[l_1..r_1], P_1$, $R_2[l_2..r_2], P_2$, and M
$S_M = \{(0,0),(i_1,j_1),...(i_K,j_K)\}$.
where $i_{k-1} < l_1 < i_k$ and $i_{l-1} < r_1 < i_l$
and $j_{k-1} < l_2 < j_k$ and $j_{l-1} < r_2 < j_l$

compute a sorted list S_1 of pairs in $S(R_1[l_1, r_1])$;
compute a sorted list S_2 of pairs in $S(R_2[l_2, r_2])$;

Let $\text{left}(S_1[i_{1_s}]) = \text{right}(S_1[i_{1_s}]) = i_{k-1+s}$, $1 \le s \le l-k$
and $\text{left}(S_2[j_{2_t}]) = \text{right}(S_2[j_{2_t}]) = j_{k-1+s}$, $1 \le t \le l-k$
Let $i_{1_0} = 0, i_{1_{l-k+1}} = |S_1| + 1$
and $j_{2_0} = 0, j_{2_{l-k+1}} = |S_2| + 1$

compute $\text{left}_1[]$ and $\text{left}_2[]$;
compute $\text{cross\_left}_1[]$ and $\text{cross\_left}_2[]$;
compute $\text{cross\_weight}_1[]$ and $\text{cross\_weight}_2[]$;

$D_1(0,0) = 0$

for $t := 0$ **to** $l-k$
  for $j := j_{2_t} + 1$ **to** $j_{2_{t+1}} - 1$
    Compute $D_1(i_{1_t}, j)$ as in Lemma 7
  for $i := i_{1_t} + 1$ **to** $i_{1_{t+1}} - 1$
    Compute $D_1(i, j_{2_t})$ as in Lemma 8
  for $i := i_{1_t} + 1$ **to** $i_{1_{t+1}} - 1$
    for $j := j_{2_t} + 1$ **to** $j_{2_{t+1}} - 1$
      Compute $D_1(i,j)$ as in Lemma 9, 10 11, 12, and 13
  $D_1(i_{1_{t+1}}, j_{2_{t+1}}) = D_1(i_{1_{t+1}} - 1, j_{2_{t+1}} - 1)$

Fig. 2. Procedure: computing $D(R_1[l_1, r_1], R_2[l_2, r_2], M)$.

Input: $R_1[1..m](P_1)$, $R_2[1..n](P_2)$, and M
$S_M = \{(0,0),(i_1,j_1),...(i_K,j_K)\}$.

Compute a sorted (by 3′ end) stem list L_1 for R_1.
Compute a sorted (by 3′ end) stem list L_2 for R_2.

for $i := 1$ **to** $|L_1|$
  for $j := 1$ **to** $|L_2|$
    let $L_1[i] = (i_1, j_1, k_1)$
    let $L_1[j] = (i_2, j_2, k_2)$
    if (i_1, j_1) and (i_2, j_2) are compatible with M
      compute $D(R_1[i_1, j_1], R_2[i_2, j_2])$

compute $D(R_1[1,m](P_1), R_2[1,n](P_2), M)$

Fig. 3. An algorithm: computing $D(R_1(P_1), R_2(P_2), M)$.

For the time complexity, the worst case would be that the constraint M is empty. The time to compute $D(R_1[l_1, r_1], R_2[l_2, r_2], M)$ is bounded by $O(|S(R_1)| \times |S(R_2)|)$. Since $|S(R_1)| < m$ and $|S(R_2)| < n$, the time complexity of the algorithm is $O(\text{stem}(R_1) \times \text{stem}(R_2) \times m \times n)$. The space complexity of the algorithm is $O(|S(R_1)| \times |S(R_2)|) = O(m \times n)$ since we only need one array to hold D_1 and another to hold D_2.

3.3. Application to real RNA data

Since the problem of computing the optimal edit distance between RNA tertiary structures (in the worst-case) is NP-hard, we restrict our attention to features commonly found in real RNA structures in order to compute an edit distance which is approximate, but close to optimal.

In the real applications, the input usually contains secondary and tertiary interactions in the form of $R(S \cup T)$, where S is a set of base pairs with no crossings forming the secondary structure of R, and T is a set of tertiary base pairs. The number of tertiary interactions, however, is always relatively small compared with the number of secondary interactions. In addition, when comparing $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$, one can assume that base pairs in S_1 should match to base pairs in S_2 and base pairs in T_1 should match to base pairs in T_2.

Using our algorithm, the following method computes the approximate edit distance when both RNAs contain tertiary interactions.

3.3.1. Method

1. Input: $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$.
2. Compute $D(R_1(S_1), R_2(S_2))$.
3. Let M be the optimal mapping from step 2. Compute $D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2), M)$.

Given $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$, we first compute $D(R_1(S_1), R_2(S_2))$. Since there is no tertiary interaction, we can compute the optimal solution and produce the optimal mapping M. We then use M as a constraint and compute $D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2), M)$.

Essentially, this method tries to find the best secondary structure matching first and ignores the tertiary interactions. Once this matching is found and fixed, we then try to add in tertiary interactions by using this matching as a constraint and put in another set of base pairs which within themselves have no crossing, but can have crossings with the constrained base pairs in M. Although this is not an optimal solution, in practice it produces a reasonable result by matching most of the base pairs.

In the following, we assume that the cost of an operation on a base pair is no less than the sum of the costs of the same operation on the two bases in the base pair. This means that given base pairs r and s, we assume that:

- $\gamma(r \to s) \geqslant \gamma(\text{left}(r) \to \text{left}(s)) + \gamma(\text{right}(r) \to \text{right}(s))$,
- $\gamma(r \to \lambda) \geqslant \gamma(\text{left}(r) \to \lambda) + \gamma(\text{right}(r) \to \lambda)$, and
- $\gamma(\lambda \to s) \geqslant \gamma(\lambda \to \text{left}(s)) + \gamma(\lambda \to \text{right}(s))$.

3.3.2. Bounds on edit distance

We now prove the following lemmas which give the bounds of our method. The optimal edit distance (the computation of which is NP-hard) is bounded below by the secondary structure matching given in [14], and bounded above by the constrained edit distance method given in this paper.

Lemma 14. *Given* $R_1(S_1 \cup T_1)$ *and* $R_2(S_2 \cup T_2)$, *then*

$$D(R_1(S_1), R_2(S_2)) \leqslant D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2)).$$

Proof. Consider the best mapping between $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$: we want to construct a mapping M between $R_1(S_1)$ and $R_2(S_2)$.

For any (r,s) where $r \notin S_1 \cup T_1$ and $s \notin S_2 \cup T_2$ (r and s are single bases), we add it to M. For any (r,s) where $r \in S_1$ and $s \in S_2$, we add it to M. For any (r,s) where $r \in T_1$ and $s \in T_2$, we can break the base pairs into single bases and put $(\text{left}(r), \text{left}(s))$ and $(\text{right}(r), \text{right}(s))$ into M.

It is clear from the construction and our assumption that $\gamma(M) \leqslant D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2))$. Therefore $D(R_1(S_1), R_2(S_2)) \leqslant D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2))$. □

Lemma 15. *Given* $R_1(S_1 \cup T_1)$ *and* $R_2(S_2 \cup T_2)$, *let* M *be the optimal mapping between* $R_1(S_1)$ *and* $R_2(S_2)$ *from the computation of* $D(R_1(S_1), R_2(S_2))$, *then*

$$D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2)) \leqslant D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2), M).$$

Proof. Trivial. □

From Lemma 14 and 15 we have the following inequality, establishing upper and lower bounds on our algorithm:

Theorem 2 (Bounds of algorithm).

$$\begin{aligned} D(R_1(S_1), R_2(S_2)) &\leqslant D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2)) \\ &\leqslant D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2), M). \end{aligned}$$

Proof. Immediate from above. □

3.3.3. Experimental results

We tested our method by performing experiments with seven RNA tertiary structures taken from the Ribonuclease P database [2] (see Appendix A). Ribonuclease P is the ribonucleoprotein endonuclease that cleaves transfer RNA precursors, removing 5′ precursor sequences and generating the mature 5′

Table 1
Experimental results: boldface numbers represent constrained edit distance, roman numbers indicate secondary structure distance

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	
R_1	**0**	232	**236** 211	**222** 282	**286** 248	**249** 216	**216** 243	**243**
R_2			**0** 207	**226** 300	**314** 194	**195** 194	**197** 241	**244**
R_3				**0** 302	**314** 210	**211** 215	**228** 215	**217**
R_4					**0** 313	**328** 294	**303** 320	**328**
R_5						**0** 193	**198** 228	**240**
R_6							**0** 225	**225**
R_7								**0**

See Appendix A for selected figures of these RNAs. R_1 – *Alcaligenes-eutrophus*; R_2 – *Anacystis-nidulans*; R_3 – *Agrobacterium-tumefaciens*; R_4 – *Bacillus-brevis*; R_5 – *Borrelia-burgdorferi*; R_6 – *Bacteroides-thetaiotaomicron*; R_7 – *Chlorobium-limicola*.

terminus of the tRNA. These seven RNAs are chosen from seven different groups of the database. Some of the RNAs are quite similar, i.e. R_2, R_5 and R_6, one is quite different, i.e. R_4, and others are in between, see Fig. 4.

We use a very simple score function, where for single bases $\gamma(a \to b) = 0$ if $a = b$, $\gamma(a \to b) = 1$ if $a \neq b$; and for base pairs, $\gamma((a,b) \to (c,d)) = 0$ if $a = c$ and $b = d$, $\gamma((a,b) \to (c,d)) = 1$ if $a = c$ and $b \neq d$ or $a \neq c$ and $b = d$, $\gamma((a,b) \to (c,d)) = 2$ if $a \neq c$ and $b \neq d$.

The results are shown in Table 1 where for each pair of RNAs, there are two numbers. The boldface number indicates the constrained edit distance measure using our method ($D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2), M)$, which is an upper bound on the optimal edit distance) and the other number indicates the lower bound ($D(R_1(S_1), R_2(S_2))$) produced by ignoring tertiary interactions, which is produced in the process of computing the constrained edit distance. In general, the upper and lower bounds are very close, which means that our method produces an edit distance which is very close to optimal, and in some cases actually reaches optimality.

4. Conclusion

We have presented a new algorithm to compute similarity measure between RNA structures. We provide a method using this algorithm which can compute the similarity measure in practical applications even when the inputs are RNA tertiary structures.

Acknowledgements

We thank anonymous referees for their helpful comments which improved the quality of this paper.

Appendix A. Selected RNA figures

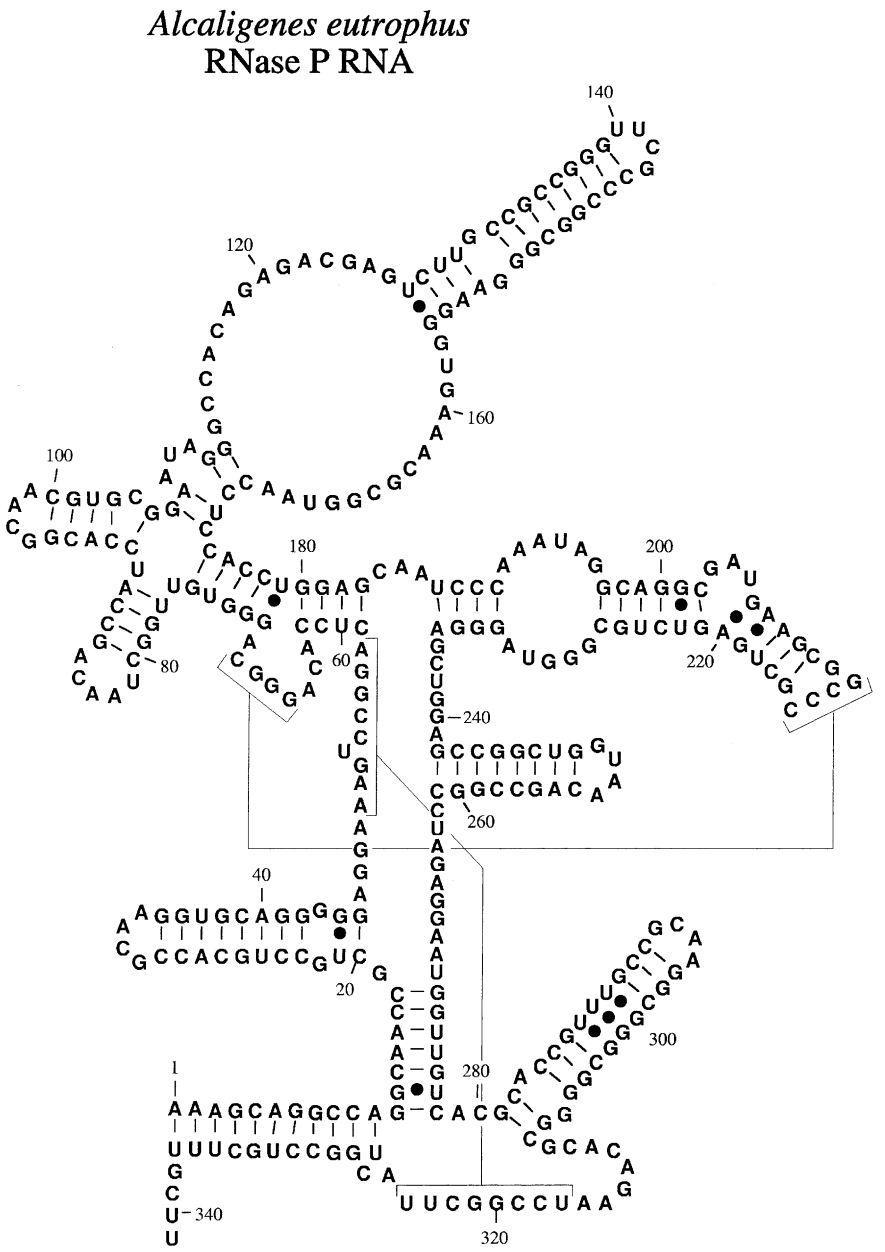

Fig. 4. Four RNAs taken from the RNase P database – in consequetive order, R_1, R_2, R_4, R_6. These images taken from http://www.mbio.ncsu.edu/RNaseP/.

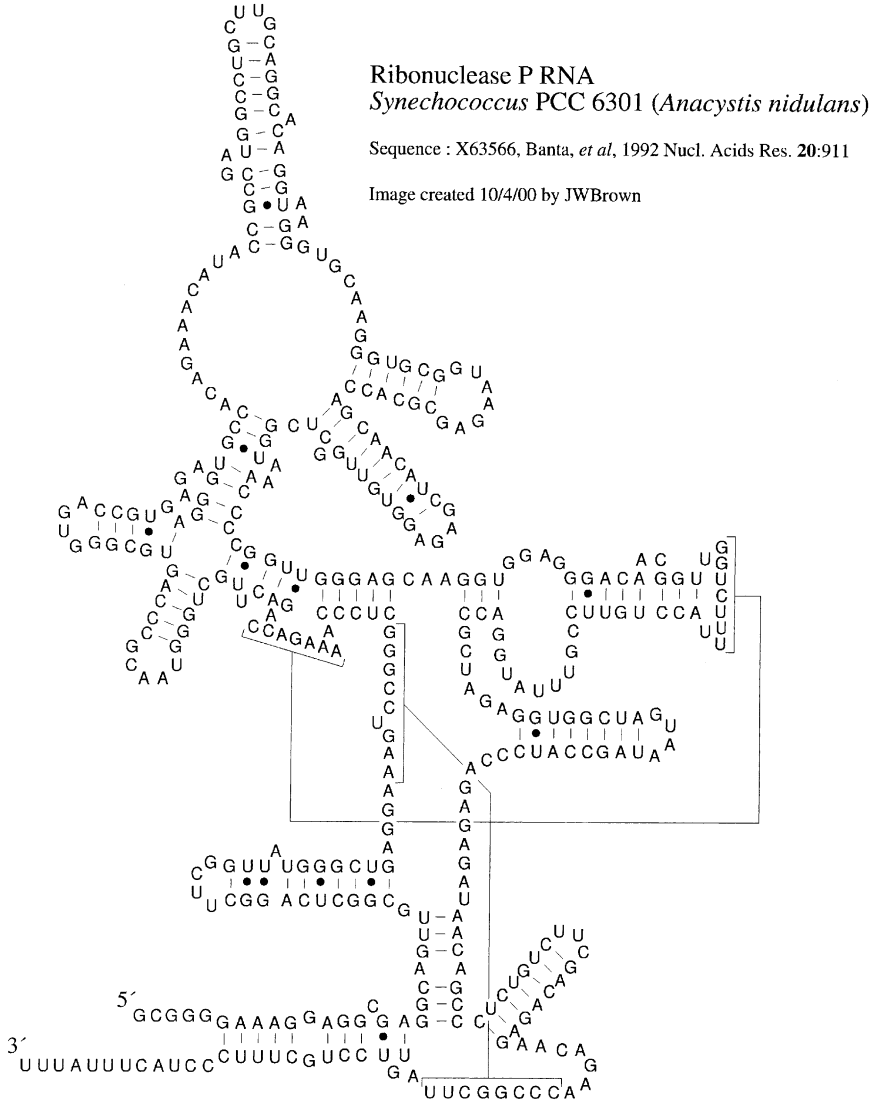

Fig. 4. (*continued*).

Bacillus brevis
RNase P RNA

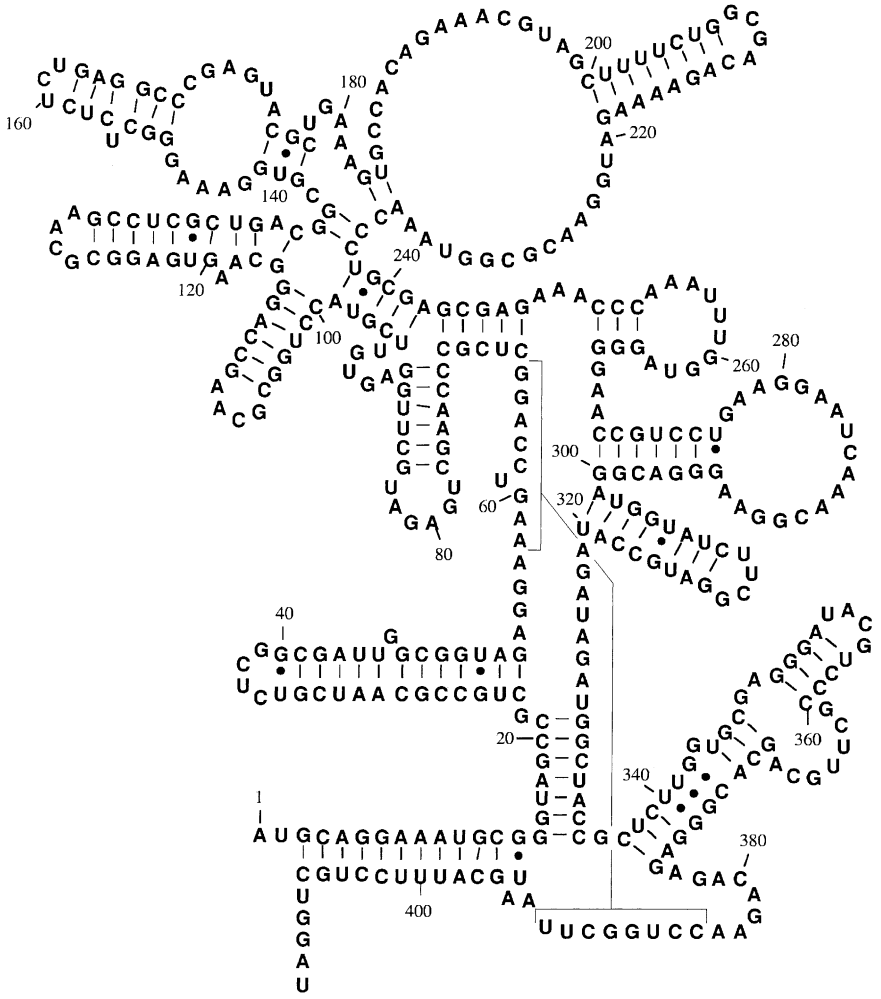

Fig. 4. (*continued*).

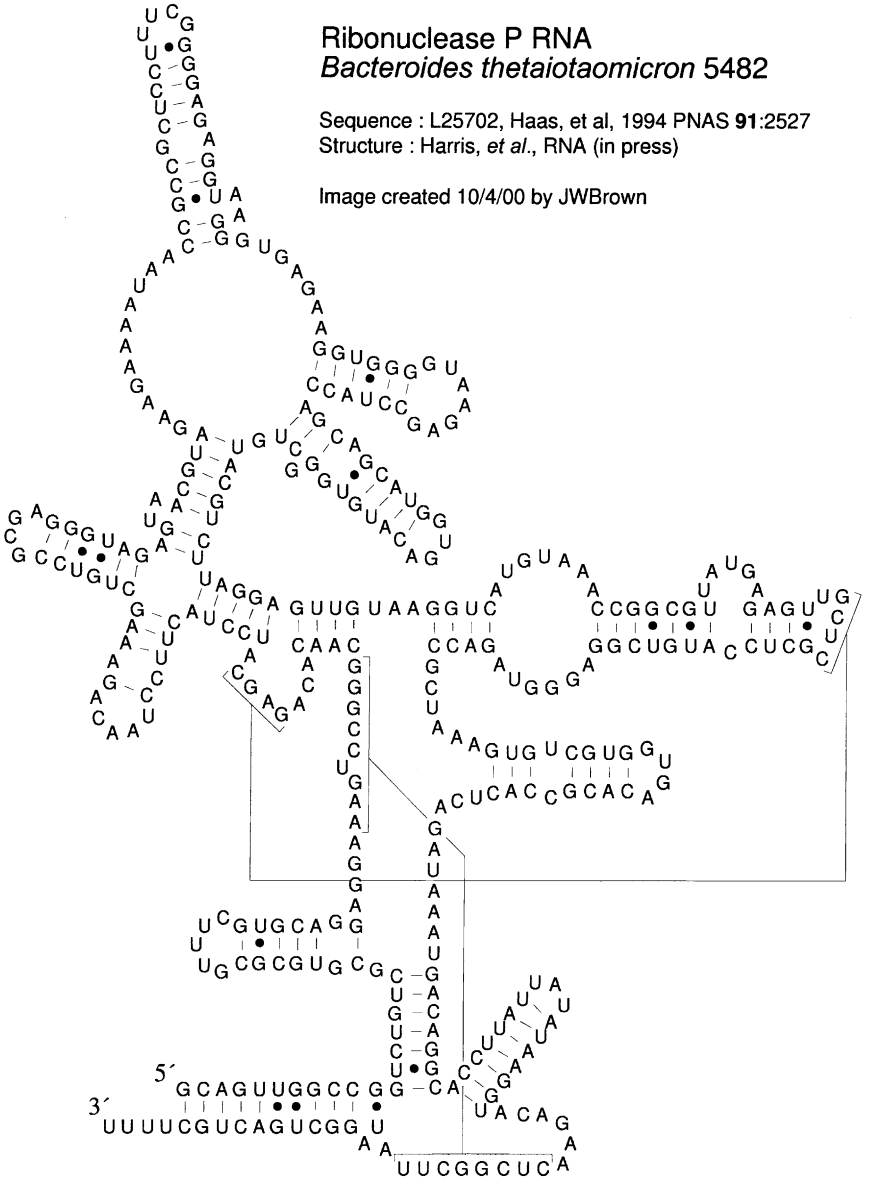

Fig. 4. (*continued*).

References

[1] V. Bafna, S. Muthukrishnan, R. Ravi, Comparing similarity between RNA strings, in: Proceedings of the Combinatorial Pattern Matching Conference '95, Lecture Notes in Computer Science, vol. 937, 1995, pp. 1–14.
[2] J.W. Brown, The ribonuclease P database, Nucleic Acids Res. 27 (1999) 314.
[3] F. Corpet, B. Michot, RNAlign program: alignment of RNA sequences using both primary and secondary structures, Comput. Appl. Biosci. 10 (4) (1995) 389–399.
[4] G. Collins, S.Y. Le, K. Zhang, A new method for computing similarity between RNA structures, in: Second International workshop on Biomolecular Informatics, Atlantic City, March, 2000.
[5] P.N. Klein, Computing the edit-distance between unrooted ordered trees, in: Proceedings of Annual European Symposium on Algorithms 98, Lecture Notes in Computer Science, vol. 1461, 1998, pp. 91–102.
[6] S.Y. Le, R. Nussinov, J.V. Mazel, Tree graphs of RNA secondary structures and their comparisons, Comput. Biomed. Res. 22 (1989) 461–473.
[7] S.Y. Le, J. Owens, R. Nussinov, J.H. Chen, B. Shapiro, J.V. Mazel, RNA secondary structures: comparisons and determination of frequently recurring substructures by consensus, Comput. Appl. Biosci. 5 (1989) 205–210.
[8] S.E. Needleman, C.D. Wunsch, A general method applicable to the search for similarities in the amino-acid sequences of two proteins, J. Mol. Bio. 48 (1970) 443–453.
[9] B. Shapiro, An algorithm for comparing multiple RNA secondary structures, Comput. Appl. Biosci. 4 (3) (1988) 387–393.
[10] B. Shapiro, K. Zhang, Comparing multiple RNA secondary structures using tree comparisons, Comput. Appl. Biosci. 6 (4) (1990) 309–318.
[11] T.F. Smith, M.S. Waterman, The identification of common molecular subsequences, J. Mol. Bio. 147 (1981) 195–197.
[12] T.F. Smith, M.S. Waterman, Comparison of biosequences, Adv. Appl. Math. 2 (1981) 482–489.
[13] K.C. Tai, The tree to tree correction problem, J. Assoc. Comput. Mach. 26 (3) (1979) 422–433.
[14] K. Zhang, Computing similarity between RNA secondary structures, in: Proceedings of IEEE International Joint Symposia on Intelligence and Systems, Rockville, MD, 1998.
[15] K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, SIAM J. Comput. 18 (6) (1989) 1245–1262.
[16] K. Zhang, L. Wang, B. Ma, Computing similarity between RNA structures, in: Proceedings of the Tenth Symposium on Combinatorial Pattern Matching, Springer-Verlag's Lecture Notes in Computer Science, vol. 1645, 1999, pp. 281–293.
[17] M. Zuker, On finding all suboptimal foldings from an RNA molecule, Science 244 (1989) 48–52.
[18] M. Zuker, D. Sankoff, RNA secondary structures and their prediction, Bull. Math. Biol. 46 (1984) 591–621.
[19] M. Zuker, P. Stiegler, Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, Nucleic Acid Res. 9 (1981) 133–148.